REMARKS

This is intended as a full and complete response to the Office Action dated October 4, 2007, having a shortened statutory period for response set to expire on January 4, 2008. Please reconsider the claims pending in the application for reasons discussed below.

Claims 1-2, 4-6, 8-12 and 14-20 are pending in the application. Claims 1-2, 4-6, 8-12 and 14-20 remain pending following entry of this response.

Claim Rejections - 35 U.S.C. § 103

Claims 1-2, 4-6, 9-12 and 15-19 are rejected under 35 U.S.C. 103(a) as being unpatentable over *Wimble* (U.S. Patent No. 5,812,850) in view of *Knouse et al.* (U.S. Publication 2007/0044144, hereinafter, "*Knouse*") and further in view of *Warmink et al.* (U.S. Patent No. 6,611,924, hereinafter, "*Warmink*"). Applicants respectfully traverse this rejection.

Regarding claims 1 and 9:

The Examiner bears the initial burden of establishing a *prima facie* case of obviousness. *See* MPEP § 2142. To establish a *prima facie* case of obviousness three basic criteria must be met. First, there must be some suggestion or motivation, either in the references themselves or in the knowledge generally available to one ordinary skill in the art, to modify the reference or to combine the reference teachings. Second, there must be a reasonable expectation of success. Third, the prior art reference (or references when combined) must teach or suggest all the claim limitations. *See* MPEP § 2143. The present rejection fails to establish at least the third criteria.

Applicants submit that *Wimble* does not disclose “a method of debugging executable code configured to access associated data in a data repository” that includes “determining whether the monitored executable code has accessed the associated data in the data repository,” as recited by claim 1. Claim 9 recites similar

limitations. Instead, *Wimble* discloses "a debugging system which provides an interactive and dynamic environment for computer program debugging." *Wimble*, 1:18-20. "The debugging system uses a database of information relating machine executable code to source code. The database is developed during the compilation process using an extensible object-oriented set of tools." *Wimble*, Abstract. The debugger disclosed in *Wimble*:

> generates *information to be used by the debugger* in a Debugging System. The information may be in the form of data bases. Debugging information is really a database of information about a compiled program.

Wimble, 6:30-34. The Examiner relies on this passage to reject claims 1 and 9. Clearly however, the passage describes a debugger configured to generate and access a database to performing debugging operations. For example, *Wimble* discloses that information in the debugger database includes "a collection of Symbolic Elements 417 which is to be built, and maintains *a description of the program once it has been built*. *Wimble*, 8:27-37. As the highlighted passages make clear, the debugger database of *Wimble* is created and used by the debugger to assist in the debugging process, and is not accessed by the program being debugged. In contrast, the present claims are directed to a debugger used to analyze "executable code configured to access associated data in a data repository."

Further, the claimed method includes a step of "determining whether the monitored executable code has accessed the associated data in the data repository." To support the present rejection, the Examiner continues to suggest:

> [*Wimble* discloses] determining whether the monitored executable code has accessed the associated data in the data repository (Col12:47-67, '... a component in the Debugger Database ...')

Office Action, p. 3. Respectfully, "a component in the debugger database" simply fails to disclose the claimed method step of "determining whether the monitored executable code has accessed the associated data in the data repository." Further, the program being debugged using the System of *Wimble* would not even access the debugging information used by the debugger. Set out in full, the cited passage provides:

ADOPTPCSOURCEMAP

FIG. 11 is a block diagram of the function AdoptPCSourceMap 129 for asking the information reader to adopt a particular format of information from a provider. The provider must put together a list of PCSourceMapElement entries 130 and ask the Debugger to adopt the information which describes a component in the Debugger Database. The following shows the Adopt function interface:

---

void AdoptPCSourceMap (const THoopsPropertyName name, TPCSourceMapList* pcSourceMap);

There is a different map kept to describe the Interface v. the Implementation properties of a component, so the developer must supply a "name" parameter to indicate which property the map describes.

THE TOKEN MAP

FIG. 12 is a block diagram showing a general overview of Token Map 66, and details of the Token Map entry objects 78.

Plainly, nothing in this passage discloses the step of determining whether the code being debugged accessed data from a data repository. Instead, the passage describes the use of function call “AdoptPCSourceMap” to ask an “information reader” to adopt a particular format of information from a provider. For all these reasons, Applicants submit that while *Wimble* discloses a debugger configured to create a database of debugging information, this fails to disclose a debugger configured for executable code that is *itself configured to access associated data in a data repository*, in the manner claimed.

Further, it directly follows that *Wimble* does not disclose the recited limitation of determining whether the monitored executable code has accessed the associated data in the data repository, as it would make no sense for the system of *Wimble* to do so. The debugger of *Wimble* would not create the debugger database and then monitor whether the very same debugger accesses the debugger database. Access by the debugger to the debugger database would be presumed. Otherwise, the debugger would not bother to create it. On this basis alone, Applicants submit that the rejection should be withdrawn and the claims be allowed.

Furthermore, claim 1 recites a limitation of:

> upon determining not to display the associated data on the basis of the referenced predefined access restriction rules, outputting masking characters on an output screen indicative of the associated data without revealing a value of the associated data, whereby selected data from the data repository is concealed from a user debugging the executable code.

In other words, if a program being debugged accesses data from a data repository that is restricted data (i.e., person debugging the program is not authorized to view certain data from the database), this information may be masked using masking characters. Claim 9 recites a similar limitation. The Examiner concedes that neither Wimble nor *Knousue* disclose this limitation but continues to suggest that *Warmink* does so. Specifically, the Examiner argues:

> *Warmink* discloses upon determining not to display the associated data on the basis of the referenced predefined access restriction rules, outputting masking(Col 2:35-67, " ... developer-selected masks ... ") characters on an output screen indicative of the associated data without revealing a value of the associated data, whereby selected data from the data repository is concealed from a user debugging the executable code(Col 2:35-67, " ... based on some specified rules of filtering ... ")

Office Action p.5. However, the "developer-selected masks" of *Warmink* completely fail to disclose the claimed limitation of "outputting masking characters on an output screen." The "developer selected masks" discussed in *Warmink* are used to allow some debugging messages to be output to a "debug port" using a unique number to represent a given debug message (thereby reducing message size) while filtering others. However, the unique number does not prevent or conceal the debugging message, in fact, the opposite is the case. *Warmink* goes on to describe how the "unique numbers" are used to generate debug messages presented to a developer when a "debug PC" is attached to a "debug port." Specifically, *Warmink* teaches:

> the "debug pc converts the debug value to a corresponding text string using a "number to string mapping table. The text string, not the ENUM value, is thus inserted in the appropriate place in the debug output string of data and displayed on monitor 126 along with other data of the debug output string.

Warmink, 7:65-67 – 8:1-3. The material cited by the Examiner refers to debugging messages that may be "selectively enabled based on developer-selected masks, based on some specified rules of filtering." *Warmink*, 2:39-40. In other words,

as used in *Warmink*, a numerical value—referred to as a "mask value"— is used as a reference to messages output during program execution. No use of masking characters is disclosed, or would be useful using this debugging technique. That is, it makes no sense whatsoever to suggest that the debugging messages, meant to inform the programmer of aspects of program operational state, could be replaced with masking characters when displayed to a user and remain useful at all. Thus, Applicants contend that the "mask value" used in *Warmink* to filter which debug messages are displayed on a "debug PC" in no way discloses the claimed method step of "outputting masking characters on an output screen indicative of the associated data without revealing a value of the associated data, whereby selected data from the data repository is concealed from a user debugging the executable code." Plainly, nothing is concealed, and no part of the debug message is the developer restricted from viewing.

Nevertheless, the Examiner suggests that *Warmink* does teach "outputting masking characters on an output screen indicative of the associated data without revealing a value of the associated data, whereby selected data from the data repository is concealed from a user debugging the executable code" at Col 2:35-67. However, the cited passage teaches that "the use of such debug trace statements allows the program developer to specify and thus enable only the subset of debug messages which are of interest." Whichever messages are output, are done so without being "masked" or otherwise concealed from the developer at all. Therefore, not only is it the developer who specifies what "messages" are of interest, but the "messages" are "debug messages". It is clear from the passage that there is no mention of any sort of filtering or "masking" of the "associated data."

Lastly, *Knouse* discloses "an access system" that "provides identity management and/or access management services for a network. An application program interface for the access system enables an application without a web agent front end to read and use contents of an existing encrypted cookie to bypass authentication and proceed to authorization." *Knouse,* Abstract. A web server authenticates users based on the user

profile. The user is either allowed access or denied access to resources through the web server. *Knouse,* ¶ 100.

The Examiner suggests that *Knouse* discloses "determining whether to display the associated data on the basis of whether the associated data is restricted data; (Col 5 paragraph 0100, "...performs the authentication by accessing attributes of the user's profile and the resource's authentication..."). However, the "associated data" referred to in Claim 1 is from a "data repository," whereas *Knouse* makes no mention of a data repository. Moreover, the determination of "whether the associated data is restricted data" occurs "during the debugging session," as recited in Claim 1. *Knouse* makes no mention of a debugging session. Therefore, *Knouse* does not teach "determining whether to display the associated data on the basis of whether the associated data is restricted data," as recited in Claim 1.

Accordingly, for all the foregoing reasons, Applicants submit that claims 1, 9, and the claims dependent therefrom, are patentable over *Wimble* in view of *Knouse* and further in view of *Warmink*.

Regarding claims 16-19:

The Examiner suggests that *Wimble*, in view of *Knouse* and further in view of *Warmink*, discloses a computer-readable medium containing a debug program which, when executed, performs an operation of debugging code configured to access associated data in a repository. However, Applicants contend that *Wimble*, in view of *Knouse* and further in view of *Warmink*, fails to disclose at least the recited limitation of "a debug engine configured to selectively pass data to the debugger user interface according to predefined access restriction rules defining at least one rule prohibiting at least a portion of the associated data from being displayed to a user operating the debug program, whereby selected data from the data repository is concealed from the user debugging the executable code."

The Examiner concedes that *Wimble* and *Knouse* fail to disclose this limitation, but asserts that *Warmink* does. For all the reasons given above, however, Applicants

submit that the passages cited from *Warmink* fail to teach or suggest a debugger interface configured to prohibit "at least a portion of the associated data from being displayed to a user operating the debug program, whereby selected data from the data repository is concealed from the user debugging the executable code." Rather, as discussed above, *Warmink* discloses a debugger user interface that allows a numerical value to filter which embedded debug messages (or codes representing such messages) are output during the execution of a program. Plainly, the debugging messages are not selected data from a data repository, as the debugging messages are embedded within the program code. Accordingly, Applicants assert that *Wimble*, in view of *Warmink*, fails to teach or suggest the limitations recited by claim 16.

Therefore, for all the foregoing reasons, claims 1, 9, 16, and the claims dependent therefrom are believed to be allowable, and allowance of these claims is respectfully requested.

Claims 8, 14 and 20 are rejected under 35 U.S.C. 103(a) as being unpatentable over *Wimble*, *Knouse* in view of *Warmink* and further in view of *Kolawa et al.* (U.S. Patent No. 6,085,029, hereinafter, "*Kolawa*").

Regarding claims 8, 14, and 20:

Applicants respectfully traverse this rejection. Claims 8, 14, and 20 depend from one of claims 1, 9 or 16, and are therefore believed to be allowable for the reasons provided above. Accordingly, withdrawal of this rejection is respectfully requested.

Therefore, for all of the foregoing reasons, the claims are believed to be allowable, and allowance of the claims is respectfully requested.

Conclusion

Having addressed all issues set out in the office action, Applicants respectfully submit that the claims are in condition for allowance and respectfully request that the claims be allowed.

Respectfully submitted, and
S-signed pursuant to 37 CFR 1.4,

/Gero G. McClellan, Reg. No. 44,227/

Gero G. McClellan
Registration No. 44,227
PATTERSON & SHERIDAN, L.L.P.
3040 Post Oak Blvd. Suite 1500
Houston, TX 77056
Telephone: (713) 623-4844
Facsimile: (713) 623-4846
Attorney for Applicants